
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

```
  SSSSSSSS  YY      YY   SSSSSSSS    SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    FFFFFFFFFF  MM      MM
  SSSSSSSS  YY      YY   SSSSSSSS    SSSSSSSS  EEEEEEEEEE  TTTTTTTTTT  PPPPPPPP    FFFFFFFFFF  MM      MM
SS          YY      YY SS          SS          EE              TT      PP      PP  FF          MMMM  MMMM
SS          YY      YY SS          SS          EE              TT      PP      PP  FF          MMMM  MMMM
SS            YY  YY   SS          SS          EE              TT      PP      PP  FF          MM  MM  MM
SS            YY  YY   SS          SS          EE              TT      PP      PP  FF          MM  MM  MM
  SSSSSS        YY       SSSSSS      SSSSSS    EEEEEEEE        TT      PPPPPPPP    FFFFFFFF    MM      MM
  SSSSSS        YY       SSSSSS      SSSSSS    EEEEEEEE        TT      PPPPPPPP    FFFFFFFF    MM      MM
        SS      YY             SS          SS  EE              TT      PP          FF          MM      MM
        SS      YY             SS          SS  EE              TT      PP          FF          MM      MM
        SS      YY             SS          SS  EE              TT      PP          FF          MM      MM
        SS      YY             SS          SS  EE              TT      PP          FF          MM      MM
SSSSSSSS        YY     SSSSSSSS    SSSSSSSS    EEEEEEEEEE      TT      PP          FF          MM      MM
SSSSSSSS        YY     SSSSSSSS    SSSSSSSS    EEEEEEEEEE      TT      PP          FF          MM      MM
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  SYSSETPFM - SET PAGE FAULT MONITORING
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27
0000    28 ;++
0000    29 ; FACILITY: Measurement System Service
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       This module enables a page fault monitoring service within the
0000    34 ;       operating system. On each page fault, the virtual address, the PC,
0000    35 ;       and the process CPU time are saved in a buffer to be output by either
0000    36 ;       a cooperating subprocess or an image-based AST routine.
0000    37 ;
0000    38 ; ENVIRONMENT: Kernel Mode
0000    39 ;
0000    40 ; AUTHOR: Henry M. Levy , CREATION DATE: 7-May-1977
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;
0000    44 ;       V03-007 SSA0026         Stan Amway      9-Jul-1984
0000    45 ;               In PFM$MON, raise IPL to IPL$_HWCLK when
0000    46 ;               doing CPU time reference update.
0000    47 ;
0000    48 ;       V03-006 SSA0025         Stan Amway      25-Jun-1984
0000    49 ;               Add global symbol PFM$C_BUFCNT for use by PCA.
0000    50 ;
0000    51 ;       V03-005 SSA0019         Stan Amway      12-Mar-1984
0000    52 ;               Prevent subprocess running PFMFILWRT from deallocating PMB.
0000    53 ;               Add access mode checking for subfunction and stop requests.
0000    54 ;
0000    55 ;       V03-004 SSA0013         Stan Amway      28-Feb-1984
0000    56 ;               Properly handle monitoring termination by
0000    57 ;               dequeueing AST block from PCB.
```

```
0000     58 ;                Ensure that PFM$PURGE will be invoked if service
0000     59 ;                is called with a stop monitoring request (notably
0000     60 ;                from SYSRUNDWN) and a PMB is still allocated.
0000     61 ;                Inhibit buffer flush if buffer contains no records.
0000     62 ;                Fix bug that inhibited the entry of a CPU timestamp
0000     63 ;                into the first page fault buffer.
0000     64 ;                Move PFM$GETBUF and ALLPMB routines to paged
0000     65 ;                PSECT.
0000     66 ;
0000     67 ;        V03-003 SSA0007         Stan Amway      2-Feb-1984
0000     68 ;                Removed restriction of 1 process per group.
0000     69 ;                Added AST interlock flag to keep the number of
0000     70 ;                ASTs delivered to a minimum.
0000     71 ;                Track changes in buffer format.
0000     72 ;
0000     73 ;        V03-002 SSA0004         Stan Amway      12-Dec-1983
0000     74 ;                Extensive changes to add support for the
0000     75 ;                Performance & Coverage Analyzer (PCA) being done by
0000     76 ;                the VMS Debug group.
0000     77 ;
0000     78 ;                Changes include:
0000     79 ;                Removed use of PMB list. Use PCB$L_PMB (new) instead.
0000     80 ;                Timestamping fault with process CPU time.
0000     81 ;                Adding support for image-based buffer handling.
0000     82 ;                Cleanup of error handling.
0000     83 ;                Optimizing main code paths for speed.
0000     84 ;
0000     85 ;        V03-001 CWH1002         CW Hobbs        1-Mar-1983
0000     86 ;                Convert to use extended pids.
0000     87 ;
0000     88 ;        02      RIH0033         R. I. HUSTVEDT  16-OCT-1979
0000     89 ;                CHANGE PCB$W_BYTCNT TO JIB$L_BYTCNT.
0000     90 ;
0000     91 ;        03      BLS0001         B. L. SCHREIBER 28-NOV-1979
0000     92 ;                CORRECT PAGE FAULT ERROR
0000     93 ;
0000     94 ;        04      BLS0002         B.L. SCHREIBER  28-JAN-1980
0000     95 ;                CORRECT ERROR IN PURGE ROUTINE.
0000     96 ;
0000     97 ;        05      BLS0003         B.L. SCHREIBER  30-JAN-1980
0000     98 ;                MORE ERRORS IN PURGE ROUTINE
0000     99 ;
0000    100 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;

                ;
                ; MACROS:
                ;

                        $ACBDEF                         ; define AST control block
                        $DYNDEF                         ; define dynamic structure types
                        $IODEF                          ; define I/O function codes
                        $IPLDEF                         ; define interrupt priority levels
                        $JIBDEF                         ; define job information block
                        $PCBDEF                         ; define process control block
                        $PHDDEF                         ; define process header
                        $PFBDEF                         ; define PFM buffer layout
                        $PMBDEF                         ; define PFM control block
                        $PQLDEF                         ; define process quota codes
                        $PRDEF                          ; define processor registers
                        $PRIDEF                         ; define priority increment classes
                        $PSLDEF                         ; define PSL fields
                        $SGNDEF                         ; define system parameters
                        $SSDEF                          ; define service status codes

                        .MACRO  $QUOTA  NAME=LISTEND,VALUE=0
                        .BYTE   PQL$_'NAME
                        .LONG   VALUE
                        .ENDM   $QUOTA

                ;
                ; EQUATED SYMBOLS:
                ;
                                                        ; Argument list offsets
00000004                PFMFLG = 4                      ; Function/subfunction flags
00000008                ASTADR = 8                      ; AST Routine Address
0000000C                ASTPRM = 12                     ; AST Parameter
00000010                ACMODE = 16                     ; AST Mode

00000005                BUFCNT = 5                      ; number of buffers to allocate
00000005                PFM$C_BUFCNT == BUFCNT          ; global definition of BUFCNT
0000020C                BUFSIZ = PFB$C_LENGTH           ; size of buffers
0000003F                MAXREC = <<BUFSIZ-PFB$B_BUFFER>/<2*4>>  ; max records per buffer
00000A7C                QUOTA_CHARGE = <<BUFCNT*BUFSIZ> + PMB$C_LENGTH> ; Amt to charge process byte quota
0000003C                FAULTVA = <4*7>+4+<7*4>         ; offset to va of faulting instruction
00000040                FAULTPC = <4*7>+8+<7*4>         ; offset to pc of faulting instruction

                ;
                ; OWN STORAGE:
                ;

                        .PSECT  YEXEPAGED,LONG

                ;
                ; Data for creation of subprocess to output filled
                ; buffers to disk file.
                ;
```

```
                                        0000   159
                                        0000   160 PFMFILWRT:                                   ; subprocess image name descriptor
                            00000018'   0000   161          .LONG   20$-10$
                            00000008'   0004   162          .LONG   10$
50 3A 4D 45 54 53 59 53 24  53 59 53    0008   163 10$:     .ASCII  /SYS$SYSTEM:PFMFILWRT.EXE/
45 58 45 2E 54 52 57 4C 49  46 4D 46    0014
                                        0020   164 20$:
                                        0020   165 PFMQUOTA:                                   ; subprocess quota name
                                        0020   166          $QUOTA  CPULM,0                    ; infinite CPU time
                                        0025   167          $QUOTA  BYTLM,1024                 ; byte limit for buffered I/O
                                        002A   168          $QUOTA  FILLM,1                    ; open file count limit
                                        002F   169          $QUOTA  PGFLQUOTA,256              ; paging file quota
                                        0034   170          $QUOTA  PRCLM,0                    ; no subprocesses
                                        0039   171          $QUOTA  TQELM,1                    ; timer queue entry
                                        003E   172          $QUOTA  LISTEND                    ; end of list
                                        0043   173 FILWRT:                                     ; subprocess process name
49 50 45 20 3C 5F 42 55 53  4D 46 50    0043   174          .ASCII  /PFMSUB_< EPID >/
                                  3E 20 44      004F
                                        0052   175 FILWRTPRV:                                  ; subprocess privilege vector
                        FFFFFFFF FFFFFFFF   0052   176          .LONG   -1,-1                      ; all privileges
```

```
                                J 5
005A    178          .SBTTL  SYSSETPFM - Initialize Page Fault Monitoring
005A    179 ;++
005A    180 ; FUNCTIONAL DESCRIPTION:
005A    181 ;
005A    182 ;       Page fault monitoring initialization.  Buffers are allocated from
005A    183 ;       the nonpaged pool and queued for use by the monitor.  In subprocess
005A    184 ;       mode, a subprocess is created which outputs buffers which have been
005A    185 ;       filled. In image-based mode, ASTs are delivered to signal full buffers.
005A    186 ;
005A    187 ;       When the process calls SETPFM to turn off monitoring, all
005A    188 ;       buffers are returned to the system and if in subprocess mode, the
005A    189 ;       subprocess is deleted.
005A    190 ;
005A    191 ;       In case of abnormal termination, the buffers are returned by SYSRUNDWN.
005A    192 ;
005A    193 ; CALLING SEQUENCE:
005A    194 ;
005A    195 ;       CALLS/CALLG
005A    196 ;
005A    197 ; INPUT PARAMETERS:
005A    198 ;
005A    199 ;        4(AP)  PFMFLG  Function/subfunction
005A    200 ;               bit   0    = off/on (0/1)
005A    201 ;               bits 1-30 = subfunction field
005A    202 ;                           bit 1 = Flush buffers
005A    203 ;               bit  31    = 0 indicates initialization call
005A    204 ;                            1 indicates subfunction call (if bit 0 = 1)
005A    205 ;
005A    206 ;        8(AP)  ASTADR  AST Routine Address
005A    207 ;                 = 0 Subprocess Mode
005A    208 ;                 <> 0 AST Routine address for image-based buffer handler
005A    209 ;
005A    210 ;       12(AP)  ASTPRM  AST parameter
005A    211 ;
005A    212 ;       16(AP)  ACMODE  Access Mode for AST delivery
005A    213 ;
005A    214 ;
005A    215 ; IMPLICIT INPUTS:
005A    216 ;
005A    217 ;       none
005A    218 ;
005A    219 ; OUTPUT PARAMETERS:
005A    220 ;
005A    221 ;       none
005A    222 ;
005A    223 ; IMPLICIT OUTPUTS:
005A    224 ;
005A    225 ;       none
005A    226 ;
005A    227 ; COMPLETION CODES:
005A    228 ;
005A    229 ;       SS$_NORMAL  - Success
005A    230 ;       SS$_EXQUOTA - A quota was exceeded while allocating buffers
005A    231 ;                     or creating the cooperating subprocess.
005A    232 ;       SS$_INSFMEM - Insufficient dynamic memory was available for buffering
005A    233 ;       SS$_PFMBSY  - Attempted to initialize page fault monitoring while
005A    234 ;                     already active.
```

```
                                                K 5
                                ;       SS$_NOPRIV  - Caller's access mode is less privileged than the mode
                                ;                     that started page fault monitoring.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;
                                ;--
                                
                                
                                        .PSECT  YEXEPAGED
                                
                                        .ENABLE         LSB
                                PFMBUSY:
50   0204 8F   3C                      MOVZWL  #SS$_PFMBSY,R0
            05   11                    BRB     10$
                                ILLSEQOP:
50   02DC 8F   3C                      MOVZWL  #SS$_ILLSEQOP,R0
                                10$:
                                        .DISABLE        LSB
                                        SETIPL  #0                      ; Restore IPL
                                        RET                             ; Return w/error in R0
                                
                                        .ENTRY  EXE$SETPFM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
54   00000000'EF   D0                   MOVL    L^SCH$GL_CURPCB,R4      ; get our PCB address
55   00000000'9F   D0                   MOVL    @#CTL$GL_PHD,R5         ; and PHD address (P1 window)
                                        SETIPL  #IPL$_ASTDEL            ; Protect access to PMB
56   011C C4   D0                       MOVL    PCB$L_PMB(R4),R6        ; Get address of allocated PMB (if any)
  53   04 AC   D0                       MOVL    PFMFLG(AP),R3           ; Get copy of PFMFLG
                                
                                ;       From this point on, IPL is at IPL$_ASTDEL and registers are as follows:
                                ;
                                ;               R3      Copy of PFMFLG
                                ;               R4      PCB Address
                                ;               R5      PHD Address (P1 window)
                                ;               R6      PMB Address or 0 (initialize request only)
                                
                                                                        ; *** Order of following tests makes
                                                                        ; implicit checks on PFM state ***
      00   E1                   BBC     #PHD$V_PFMFLG,-                 ; BR if monitoring not initialized
58 36 A5                                PHD$W_FLAGS(R5),START
   37 53   E9                   BLBC    R3,STOP_VEC                     ; BR if termination request
      D1   18                   BGEQ    ILLSEQOP                        ; BR if R3 >= 0 (not a subfunction call)
                                
                                ;       Subfunction Processing
                                ;
                                SUBFUNC:
      1A   10                   BSBB    CHECK_ACMODE                    ; If caller does not have privilege,
   29 50   E9                   BLBC    R0,NOPRIV                       ; return with error status
06 53   01   E1                 BBC     #1,R3,10$                       ; If flush buffer request,
00000065'EF   16                JSB     FLUSH_BUFFER                    ; do it now
                                                                        ; (R2, R4, R5 DESTROYED)
                                        ASSUME  PMB$V_MODE EQ 0
```

```
                     009F    292          ASSUME  PMB$K_IMAGE EQ 1
02 0B A6    E9       009F    293 10$:     BLBC    PMB$B_FLAGS(R6),20$     ; If image-based mode,
       71   10       00A3    294          BSBB    SET_ASTMODE             ; update AST parameters
                     00A5    295 20$:     SETIPL  #0                      ; Lower IPL
 50    01   3C       00A8    296          MOVZWL  #SS$_NORMAL,R0          ; and return with success
            04       00AB    297          RET
                     00AC    298
                     00AC    299                                          ; Determine if caller is
                     00AC    300                                          ; permitted to execute function
                     00AC    301 CHECK_ACMODE:
       50    D4       00AC    302          CLRL    R0                      ; Assume function not allowed to caller
       51    DC       00AE    303          MOVPSL  R1                      ; R1 <= PSL
    02 16    EF       00B0    304          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,- ; R1 <= caller's mode
    51 51             00B3    305                  R1,R1
2E A6 51    91       00B5    306          CMPB    R1,PMB$B_ACMODE(R6)     ; If caller less privileged than owner,
       02   1A       00B9    307          BGTRU   10$                     ; return with no access indication in R0
       50   D6       00BB    308          INCL    R0                      ; else return access OK
            05       00BD    309 10$:     RSB
                     00BE    310
                     00BE    311 NOPRIV:
 50    24   3C       00BE    312          MOVZWL  #SS$_NOPRIV,R0          ; Return w/no privilege error
                     00C1    313          SETIPL  #0
            04       00C4    314          RET
                     00C5    315
                     00C5    316 STOP_VEC:
     0142   31       00C5    317          BRW     STOP
                     00C8    318 PURGE_EXIT_R0_VEC:
     0161   31       00C8    319          BRW     PURGE_EXIT_R0
                     00CB    320 PURGE_EXIT_VEC:
     015B   31       00CB    321          BRW     PURGE_EXIT
                     00CE    322
                     00CE    323 CHECK_STOP:
 90 53      E8       00CE    324          BLBS    R3,ILLSEQOP             ; Error if not termination request
       56   D5       00D1    325          TSTL    R6                      ; If PMB doesn't exist,
       8C   13       00D3    326          BEQL    ILLSEQOP                ; don't proceed
       00   E0       00D5    327          BBS     #PMB$V_MODE,-           ; If image mode, continue normally
EB 0B A6            00D7    328                  PMB$B_FLAGS(R6),STOP_VEC
   34 A6    D1       00DA    329          CMPL    PMB$L_EPID(R6),-        ; Don't allow PFM rundown
   64 A4             00DD    330                  PCB$L_EPID(R4)          ; if the current process is the
       80   13       00DF    331          BEQL    ILLSEQOP                ; subprocess running PFMFILWRT
       E2   11       00E1    332          BRB     STOP_VEC                ; Otherwise, join STOP monitoring code
                     00E3    333 ;
                     00E3    334 ;       Initialize page fault monitoring
                     00E3    335 ;
                     00E3    336
                     00E3    337 START:
01     53    D1       00E3    338          CMPL    R3,#1                   ; Valid request ?
       E6   12       00E6    339          BNEQ    CHECK_STOP              ; BR if not
       56   D5       00E8    340          TSTL    R6                      ; PMB allocated ?
       03   13       00EA    341          BEQL    10$
    FF6B    31       00EC    342          BRW     PFMBUSY                 ; Yes - can't proceed
    0249    30       00EF    343 10$:     BSBW    ALLPMB                  ; Allocate PMB
                     00F2    344          SETIPL  #0                      ; It is now safe to lower IPL
 D0 50      E9       00F5    345          BLBC    R0,PURGE_EXIT_R0_VEC    ; If PMB allocation error, quit now
       51   DC       00F8    346          MOVPSL  R1                      ; Store caller's access mode
02     16   EF       00FA    347          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,-
51     51             00FD    348                  R1,R1                   ; as owner's access mode
```

```
                 2E A6    51   90  00FF   349          MOVB    R1,PMB$B_ACMODE(R6)     ; for subsequent privilege checking
                    08 AC    D5  0103   350          TSTL    ASTADR(AP)              ; Are we initializing image-based mode ?
                       28    13  0106   351          BEQL    SET_SUBPMODE            ; If no, setup subprocess mode
                       0C    10  0108   352          BSBB    SET_ASTMODE             ; Else setup image-based AST mode
                             010A   353          ASSUME  PMB$V_MODE EQ 0
                             010A   354          ASSUME  PMB$K_IMAGE EQ 1
                    0B A6    96  010A   355          INCB    PMB$B_FLAGS(R6)         ; Set image-based AST mode flag
                             010D   356  START_OK:
                       00    E2  010D   357          BBSS    #PHD$V_PFMFLG,-         ; Enable page fault monitoring
                 00 36 A5          010F   358                  PHD$W_FLAGS(R5),10$
                    50    01    3C  0112   359  10$:    MOVZWL  #SS$_NORMAL,R0          ; and return with success status
                                04  0115   360          RET
                                    0116   361
                                    0116   362  ;
                                    0116   363  ;       Setup image-based mode AST parameters
                                    0116   364  ;
                                    0116   365
                                    0116   366  SET_ASTMODE:
                                    0116   367          ASSUME  PMB$L_ASTPRM EQ PMB$L_AST+4
                                    0116   368          ASSUME  ASTPRM EQ ASTADR+4
                       34 A6    08 AC  7D  0116   369          MOVQ    ASTADR(AP),PMB$L_AST(R6); Save AST address and parameter
                 50    10 AC    02  00  EF  011B   370          EXTZV   #0,#2,ACMODE(AP),R0     ; R0 <= request AST delivery mode
                                FEDC'  30  0121   371          BSBW    EXE$MAXACMODE           ; Maximize requested and allowable
                                   30  89  0124   372          BISB3   #<ACB$M_NODELETE!ACB$M_PKAST>,- ; access modes & store with
                             2F A6    50  0126   373                  R0,PMB$B_RMOD(R6)       ; nodelete, pkast set;quota, kast, clear
                          3C A6   02D1'CF  9E  0129   374          MOVAB   W^PFM_PKAST,PMB$L_KAST(R6) ; Set piggy-back kernel AST address
                                       05  012F   375          RSB
                                    0130   376
                                    0130   377  ;
                                    0130   378  ;       Setup subprocess mode
                                    0130   379  ;
                                    0130   380
                                    0130   381  SET_SUBPMODE:                           ; (R0-R3 DESTROYED)
                                    0130   382
                          01    DD  0130   383          PUSHL   #SS$_NORMAL             ; Save room for status on stack
                                    0132   384  ;
                                    0132   385  ; Create a termination mailbox for the subprocess.
                                    0132   386  ;
                                    0132   387
                                    0132   388          $CREMBX_S CHAN=PMB$W_MBXCHN(R6),MAXMSG=#120,-
                                    0132   389                  BUFQUO=#120,PROMSK=#0
                                    014E   390
                 6E    50    D0  014E   391          MOVL    R0,(SP)
                    03 50    E8  0151   392          BLBS    R0,5$
                       00D2    31  0154   393          BRW     PURGE_EXIT              ; exit on error
                 5E    10    C2  0157   394  5$:     SUBL2   #16,SP                  ; buffer space for GETCHN on stack
                       5E    DD  015A   395          PUSHL   SP                      ; build descriptor for buffer
                       10    DD  015C   396          PUSHL   #16                     ; length of buffer
                 52    6E    DE  015E   397          MOVAL   (SP),R2                 ; get descriptor address
                                    0161   398          $GETCHN_S CHAN=PMB$W_MBXCHN(R6),PRIBUF=(R2) ; get mailbox unit #
                                    0174   399
                                    0174   400  ;
                                    0174   401  ; Form unique subprocess name using EPID of this process
                                    0174   402  ; (NOTE: The following code adds a NET 4 bytes to the stack local storage)
                                    0174   403  ;
                                    0174   404
                          3C    BB  0174   405          PUSHR   #^M<R2,R3,R4,R5>
```

```
                                                                N 5
04 A2   FEC8 CF    0F   28  0176   406          MOVC3   #15,W^FILWRT,4(R2)      ; Move name template to stack storage
                    3C   BA  017D   407          POPR    #^M<R2,R3,R4,R5>
                         017F   408                                             ; Convert hex EPID to ASCII
            51   64 A4   D0  017F   409          MOVL    PCB$L_EPID(R4),R1       ; R1 <= EPID to be converted
            57   0B A2   9E  0183   410          MOVAB   11(R2),R7               ; R7 <= address of 1st character
                 1C   58   D0  0187   411          MOVL    #28,R8                  ; R8 <= position of 1st nibble
      50   51   04   58   EF  018A   412 10$:     EXTZV   R8,#4,R1,R0             ; Get next four bits of value
   87   00000000'9F40   90  018F   413          MOVB    G^EXE$AB_HEXTAB[R0],(R7)+ ;Convert to ASCII and store
FFEC 58    FC 8F   00   9D  0197   414          ACBB    #0,#-4,R8,10$           ; Loop until all digits converted
            62   04 A2   9E  019E   415          MOVAB   4(R2),(R2)              ; Form string descriptor
                 0F   DD  01A2   416          PUSHL   #15                     ; for resultant name
                      01A4   417 ;
                      01A4   418 ; Create subprocess with high priority, full privilege and termination mailbox
                      01A4   419 ;
                      01A4   420
                      01A4   421          $CREPRC_S       PIDADR=PMB$L_EPID(R6),- ; pid of created process
                      01A4   422                          IMAGE=PFMFILWRT,-
                      01A4   423                          PRCNAM=-4(R2),-
                      01A4   424                          PRVADR=FILWRTPRV,-
                      01A4   425                          BASPRI=#6,-
                      01A4   426                          QUOTA=PFMQUOTA,-
                      01A4   427                          MBXUNT=8+12(R2),- ; unit from get channel information
                      01A4   428                          STSFLG=#4       ; disable swapping
            5E   1C   C0  01CF   429          ADDL2   #<24+4>,SP              ; Return stack local storage
            6E   50   D0  01D2   430          MOVL    R0,(SP)
               03 50   E8  01D5   431          BLBS    R0,15$
                  0082 31  01D8   432          BRW     DASSGN_EXIT             ; exit on error
            7E   54   7D  01DB   433 15$:     MOVQ    R4,-(SP)                ; Save R4, R5
                      01DE   434          SETIPL  B^20$                   ; Synchronize access to system database
  50   34 A6   D0  01E2   435          MOVL    PMB$L_EPID(R6),R0       ; Convert EPID to IPID
00000000'EF   16  01E6   436          JSB     EXE$EPID_TO_IPID
  30 A6   50   D0  01EC   437          MOVL    R0,PMB$L_PID(R6)        ; and save in PMB
00000000'EF   16  01F0   438          JSB     EXE$IPID_TO_PCB         ; Convert IPID to PCB address
011C C0   56   D0  01F6   439          MOVL    R6,PCB$L_PMB(R0)        ; Insert PMB address in subprocess PCB
                  01FB   440          SETIPL  #0
      54   8E   7D  01FE   441          MOVQ    (SP)+,R4                ; Restore R4, R5
           8E   D5  0201   442          TSTL    (SP)+                   ; Discard stacked status
         FF07   31  0203   443          BRW     START_OK                ; and take successful exit path
                    0206   444
         00000008 0206   445 20$:     .LONG   IPL$_SYNCH
                    020A   446
                    020A   447 ;
                    020A   448 ;       STOP - Termination Processing
                    020A   449 ;
                    020A   450
                    020A   451 STOP:
         FE9F   30  020A   452          BSBW    CHECK_ACMODE            ; If caller does not have privilege,
      03 50   E8  020D   453          BLBS    R0,5$
         FEAB   31  0210   454          BRW     NOPRIV                  ; return with error status
00 36 A5   00   E5  0213   455 5$:      BBCC    #PHD$V_PFMFLG,PHD$W_FLAGS(R5),10$ ; Turn off active monitoring
00000065'EF   16  0218   456 10$:     JSB     FLUSH_BUFFER            ; Flush outstanding buffers
                    021E   457                                          ; (R2, R4, R5 DESTROYED)
                    021E   458          SETIPL  #0                      ; return to IPL 0
            01   DD  0221   459          PUSHL   #SS$_NORMAL             ; Assume success
                    0223   460          ASSUME  PMB$V_MODE EQ 0
                    0223   461          ASSUME  PMB$K_IMAGE EQ 1
   02 0B A6   E8  0223   462          BLBS    PMB$B_FLAGS(R6),PURGE_EXIT ; If subprocess mode,
```

```
                06   11  0227   463          BRB     STOP_PROCESS            ; do orderly shutdown of subprocess
                         0229   464 PURGE_EXIT:                              ; (Assumes IPL=0, (SP) = status)
                50 8ED0  0229   465          POPL    R0
                         022C   466 PURGE_EXIT_R0:
                3C   10  022C   467          BSBB    PFM$PURGE               ; Return process buffers to system
                     04  022E   468          RET                             ; Return w/status in R0
                         022F   469
                         022F   470 ;
                         022F   471 ;       Orderly shutdown of subprocess
                         022F   472 ;
                         022F   473
                         022F   474 STOP_PROCESS:
                         022F   475          $FORCEX_S PIDADR=PMB$L_EPID(R6) ; force subprocess to exit
              1D 50   E9 023D   476          BLBC    R0,DASSGN_EXIT          ; don't wait for message if force failed
                         0240   477          $QIOW_S CHAN=PMB$W_MBXCHN(R6),FUNC=#IO$_READVBLK,-
                         0240   478                  P1=PMB$L_EPID(R6),-     ; buffer four bytes of termination
                         0240   479                  P2=#4                   ; ...message in unneeded PID slot
                         025D   480 DASSGN_EXIT:
                         025D   481          $DASSGN_S CHAN=PMB$W_MBXCHN(R6) ; release mailbox
                   BF    11 0268   482          BRB     PURGE_EXIT
                         026A   483
```

```
                                                C 6
                     026A   485          .SBTTL  PFM$PURGE - Return all process buffers to pool
                     026A   486
                     026A   487 ;++
                     026A   488 ;
                     026A   489 ; FUNCTIONAL DESCRIPTION:
                     026A   490 ;
                     026A   491 ;       All data buffers and the control block are returned to pool.
                     026A   492 ;
                     026A   493 ; CALLING SEQUENCE:
                     026A   494 ;
                     026A   495 ;       BSB/JSB PFM$PURGE
                     026A   496 ;
                     026A   497 ; INPUTS:
                     026A   498 ;
                     026A   499 ;       none
                     026A   500 ;
                     026A   501 ; IMPLICIT INPUTS:
                     026A   502 ;
                     026A   503 ;       PCB Address
                     026A   504 ;
                     026A   505 ; OUTPUTS:
                     026A   506 ;
                     026A   507 ;       none
                     026A   508 ;
                     026A   509 ; IMPLICIT OUTPUTS:
                     026A   510 ;
                     026A   511 ;       none
                     026A   512 ;
                     026A   513 ; SIDE EFFECTS:
                     026A   514 ;
                     026A   515 ;       none - all registers preserved
                     026A   516 ;
                     026A   517 ; ROUTINE VALUE:
                     026A   518 ;
                     026A   519 ;       none
                     026A   520 ;
                     026A   521 ; ENVIRONMENT:
                     026A   522 ;
                     026A   523 ;       Kernel mode, IPL <= IPL$_ASTDEL
                     026A   524 ;--
                     026A   525
                     026A   526
                     026A   527 PFM$PURGE::
      007F 8F    BB  026A   528          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6> ; save registers
54  00000000'EF    D0  026E   529          MOVL    L^SCH$GL_CURPCB,R4       ; get our PCB address
                       0275   530          DSBINT  #IPL$_ASTDEL             ; protect manipulation of PMB
 56   011C C4    D0  027B   531          MOVL    PCB$L_PMB(R4),R6         ; get PMB address (if any)
      011C C4    D4  0280   532          CLRL    PCB$L_PMB(R4)            ; clear PMB pointer
           56    D5  0284   533          TSTL    R6                       ; was there a PMB
           37    13  0286   534          BEQL    40$                      ; exit if not
02 0B A6    02    E5  0288   535          BBCC    #PMB$V_QAST,PMB$B_FLAGS(R6),5$ ; Is ACB enqueued on PCB ?
           47    10  028D   536          BSBB    DEQUEUE_ACB              ; Yes, remove it.
           66    D5  028F   537 5$:       TSTL    PMB$L_CURBUF(R6)         ; check if current buffer there
           06    13  0291   538          BEQL    10$                      ; branch if none there
 50    04 A6    D0  0293   539          MOVL    PMB$L_BUFBASE(R6),R0     ; else set buffer address
           2E    10  0297   540          BSBB    50$                      ; return to system
 50    14 B6    0F  0299   541 10$:      REMQUE  @PMB$Q_HDR(R6),R0        ; remove buffer from queue
```

```
                                                     D 6
                04   1D   029D    542                BVS     20$                             ; exit if none there
                26   10   029F    543                BSBB    50$                             ; return buffer
                F6   11   02A1    544                BRB     10$                             ; go back for more buffers
          50    1C B6    0F   02A3    545 20$:       REMQUE  @PMB$Q_SBPHDR(R6),R0            ; check that subprocess queue is clear
                04   1D   02A7    546                BVS     30$                             ; exit if no entry
                1C   10   02A9    547                BSBB    50$                             ; else return buffer to system pool
                F6   11   02AB    548                BRB     20$                             ; check for any more
             50 56   D0   02AD    549 30$:       MOVL    R6,R0                           ; get back PMB block address
                15   10   02B0    550                BSBB    50$                             ; deallocate control block
          50 0080 C4   D0   02B2    551                MOVL    PCB$L_JIB(R4),R0                ; Get JIB address
    20 A0 00000A7C 8F   C0   02B7    552                ADDL2   #QUOTA_CHARGE,JIB$L_BYTCNT(R0) ; Return quota to process
                     02BF    553 40$:       ENBINT                                  ; restore IPL
             007F 8F   BA   02C2    554                POPR    #^M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
                05   02C6    555                RSB                                     ; return to caller
                     02C7    556
                     02C7    557 ;
                     02C7    558 ; Return buffer to pool.
                     02C7    559 ;
                     02C7    560
          0A A0      B4   02C7    561 50$:       CLRW    PMB$W_SIZE+2(R0)        ; Clear type field for EXE$DEANONPAGED
    00000000'EF      16   02CA    562                JSB     EXE$DEANONPAGED         ; deallocate memory
                05   02D0    563                RSB
                     02D1    564
                     02D1    565 ;
                     02D1    566 ; The sole purpose of this routine is to mark the ACB as not queued.
                     02D1    567 ; It is called as a piggy-back kernel AST routine to effectively interlock the
                     02D1    568 ; operation with the monitoring termination code, which will conditionally
                     02D1    569 ; remove the ACB from the PCB.
                     02D1    570 ;
                     02D1    571 ; Inputs: R5 = ACB address, IPL = IPL$_ASTDEL
                     02D1    572 ; Outputs: PMB$V_QAST cleared, all registers preserved
                     02D1    573 ;
                     02D1    574
                     02D1    575 PFM_PKAST:
 E7 A5      04   8A   02D1    576                BICB2   #PMB$M_QAST,PMB$B_FLAGS-PMB$L_ASTQFL(R5) ; Show ACB dequeued
                05   02D5    577                RSB
                     02D6    578
                     02D6    579 ;
                     02D6    580 ; This routine is called from PFM$PURGE to dequeue the ACB
                     02D6    581 ; if PMB$V_QAST in PMB$B_FLAGS was set.
                     02D6    582 ;
                     02D6    583 ; Inputs:  R6 = PMB address, IPL=IPL$_ASTDEL,
                     02D6    584 ;                PMB$V_QAST checked and cleared by caller
                     02D6    585 ; Outputs: ACB dequeued, R5 destroyed
                     02D6    586 ;
                     02D6    587
                     02D6    588 DEQUEUE_ACB:
  55   24 A6      9E   02D6    589                MOVAB   PMB$L_ASTQFL(R6),R5     ; R5 <= address of embedded ACB
                     02DA    590                SETIPL  B^10$
00000000'EF      16   02DE    591                JSB     SCH$REMOVACB            ; Remove ACB from PCB
                     02E4    592                SETIPL  #IPL$_ASTDEL
                05   02E7    593                RSB
          00000008   02E8    594 10$:       .LONG   IPL$_SYNCH
                     02EC    595
```

```
02EC    597          .SBTTL  PFM$GETBUF - Return PFM Buffer to Caller
02EC    598
02EC    599 ;++
02EC    600 ;
02EC    601 ; FUNCTIONAL DESCRIPTION:
02EC    602 ;
02EC    603 ;       Returns a filled PFM buffer to the caller.
02EC    604 ;       For efficiency, this routine and FLUSH_BUFFER assume
02EC    605 ;       that once the collection process is awakened, it will
02EC    606 ;       continue calling PFM$GETBUF until SS$_NODATA is returned, or
02EC    607 ;       an error is encountered.
02EC    608 ;
02EC    609 ; CALLING SEQUENCE:
02EC    610 ;
02EC    611 ;       BSB/JSB PFM$GETBUF
02EC    612 ;
02EC    613 ; INPUTS:
02EC    614 ;
02EC    615 ;       R1 = Buffer address
02EC    616 ;       R2 = Buffer size
02EC    617 ;            (Should be = PFB$S_USER_BUFFER, with which it is minimized)
02EC    618 ;
02EC    619 ;       NB: No checks are made for buffer accessability.
02EC    620 ;
02EC    621 ; IMPLICIT INPUTS:
02EC    622 ;
02EC    623 ;       SCH$GL_CURPCB - PCB address of current process
02EC    624 ;       PCB$L_PMB in PCB - pointer to PMB
02EC    625 ;
02EC    626 ; OUTPUTS:
02EC    627 ;
02EC    628 ;       If R0=SS$_NORMAL or SS$_BUFFEROVF, buffer is filled with page fault data
02EC    629 ;
02EC    630 ; IMPLICIT OUTPUTS:
02EC    631 ;
02EC    632 ;       none
02EC    633 ;
02EC    634 ; ROUTINE VALUE:
02EC    635 ;
02EC    636 ;       R0 = Status
02EC    637 ;
02EC    638 ;            SS$_NORMAL
02EC    639 ;            SS$_NODATA
02EC    640 ;            SS$_ILLSEQOP
02EC    641 ;            SS$_BUFFEROVF
02EC    642 ;
02EC    643 ; SIDE EFFECTS:
02EC    644 ;
02EC    645 ;       R1-R5 destroyed
02EC    646 ;
02EC    647 ; ENVIRONMENT:
02EC    648 ;
02EC    649 ;       Kernel mode, IPL 0
02EC    650 ;--
02EC    651
02EC    652 ; NB: This code assumes that IPL synchronization (above ASTDEL) is
02EC    653 ;     not required because
```

```
                                02EC   654 ;       a) a command/response protocol is used between the
                                02EC   655 ;          main and sub processes
                                02EC   656 ;       b) SYSRUNDWN will unconditionally call SETPFM to
                                02EC   657 ;          turn off monitoring (i.e., assumption a) cannot be breached)
                                02EC   658 ;       c) no code outside of this module accesses PFM data structures
                                02EC   659
                                02EC   660 PFM$GETBUF::
54   00000000'EF    D0          02EC   661          MOVL    L^SCH$GL_CURPCB,R4      ; Get PCB address for this process
     54    011C C4  D0          02F3   662          MOVL    PCB$L_PMB(R4),R4        ; Get PMB address
                3A  13          02F8   663          BEQL    50$                     ; BR if none
                                02FA   664 10$:     SETIPL  #IPL$_ASTDEL            ; Protect buffer handling
      55    1C B4   0F          02FD   665          REMQUE  @PMB$Q_SBPHDR(R4),R5    ; Dequeue filled buffer
                26  1D          0301   666          BVS     40$                     ; Exit if none
          50  01   3C          0303   667          MOVZWL  #SS$_NORMAL,R0          ; Assume adequate buffer size
     53   0200 8F   3C          0306   668          MOVZWL  #PFB$S_USER_BUFFER,R3   ; and set number of bytes to move
          53  52   D1          030B   669          CMPL    R2,R3                   ; Is buffer big enough ?
              08   18          030E   670          BGEQ    15$                     ; BR if yes
     50   0601 8F   3C          0310   671          MOVZWL  #SS$_BUFFEROVF,R0       ; Indicate data loss
          53  52   D0          0315   672          MOVL    R2,R3                   ; Adjust number of bytes to move
              31   BB          0318   673 15$:     PUSHR   #^M<R0,R4,R5>           ; Save regs and status
61    0C A5  53   28          031A   674          MOVC3   R3,PFB$B_USER_BUFFER(R5),(R1) ; Move buffer
              31   BA          031F   675          POPR    #^M<R0,R4,R5>           ; Restore regs and status
      14 A4   65   0E          0321   676          INSQUE  (R5),PMB$Q_HDR(R4)      ; Return buffer to free list
                               0325   677 20$:     SETIPL  #0                      ; Restore IPL
                05             0328   678 30$:     RSB                             ; Return to caller
                               0329   679
     50   01AC 8F   3C          0329   680 40$:     MOVZWL  #SS$_NODATA,R0
       0B A4   02   8A          032E   681          BICB2   #PMB$M_ASTIP,PMB$B_FLAGS(R4) ; Show no AST in progress
                F1  11          0332   682          BRB     20$
     50   02DC 8F   3C          0334   683 50$:     MOVZWL  #SS$_ILLSEQOP,R0
                ED  11          0339   684          BRB     30$
                               033B   685
```

```
                     033B   687          .SBTTL  ALLPMB - Allocate PMB control block and data buffers
                     033B   688
                     033B   689 ;++
                     033B   690 ;
                     033B   691 ; FUNCTIONAL DESCRIPTION:
                     033B   692 ;
                     033B   693 ;       Allocates all process structures needed for page fault monitoring.
                     033B   694 ;
                     033B   695 ; CALLING SEQUENCE:
                     033B   696 ;
                     033B   697 ;       JSB/BSB ALLPMB
                     033B   698 ;
                     033B   699 ; INPUTS:
                     033B   700 ;
                     033B   701 ;       R4      PCB address
                     033B   702 ;       R5      PHD address
                     033B   703 ;
                     033B   704 ;
                     033B   705 ; IMPLICIT INPUTS:
                     033B   706 ;
                     033B   707 ;       none
                     033B   708 ;
                     033B   709 ; OUTPUTS:
                     033B   710 ;
                     033B   711 ;       R6 = PMB address
                     033B   712 ;
                     033B   713 ; IMPLICIT OUTPUTS:
                     033B   714 ;
                     033B   715 ;       PCB$L_PMB contains PMB address
                     033B   716 ;
                     033B   717 ; ROUTINE VALUE:
                     033B   718 ;
                     033B   719 ;       R0 = SS$_NORMAL
                     033B   720 ;          = Pool allocation error (SS$_INSFMEM, etc.)
                     033B   721 ;
                     033B   722 ; SIDE EFFECTS:
                     033B   723 ;
                     033B   724 ;       R0,R1,R2,R3,R8,R9 destroyed
                     033B   725 ;
                     033B   726 ; ENVIRONMENT:
                     033B   727 ;
                     033B   728 ;       Kernel mode, IPL = IPL$_ASTDEL
                     033B   729 ;--
                     033B   730
                     033B   731 ALLPMB:
                 1C  DD  033B   732          PUSHL   #SS$_EXQUOTA            ; stack no quota error code
                     033D   733
                     033D   734 ;
                     033D   735 ; Process should have enough quota for data buffers and PMB
                     033D   736 ;
                     033D   737
              58 0080 C4  D0 033D   738          MOVL    PCB$L_JIB(R4),R8        ; get JIB address
20 A8 00000A7C 8F  D1 0342   739          CMPL    #QUOTA_CHARGE,JIB$L_BYTCNT(R8) ; quota left?
                 6F  1A 034A   740          BGTRU   20$                     ; error if not
                     034C   741
                     034C   742 ;
                     034C   743 ; Allocate PMB control block and insert address in PMB list
```

```
                                034C   744 ;
                                034C   745
                     6E   0124 8F   3C  034C   746          MOVZWL  #SS$_INSFMEM,(SP)          ; assume memory not available
                     51   0040 8F   3C  0351   747          MOVZWL  #PMB$C_LENGTH,R1           ; get length of PMB block to allocate
                      00000000'EF   16  0356   748          JSB     EXE$ALLOCBUF               ; allocate control block
                             50   E9  035C   749          BLBC    R0,20$                     ; check that memory available
                20 A8  00000A7C 8F   C2  035F   750          SUBL2   #QUOTA_CHARGE,JIB$L_BYTCNT(R8) ; adjust quota
                             56   52  0367   751          MOVL    R2,R6                      ; copy PMB block address
                        011C C4   56   D0  036A   752          MOVL    R6,PCB$L_PMB(R4)           ; insert PMB address in PCB
                                           036F   753                                             ; N.B.: PFM$PURGE assumes that quota has
                                           036F   754                                             ; been charged if PCB$L_PMB <> 0
                                           036F   755
                                           036F   756 ;
                                           036F   757 ; Initialize PMB and allocate and queue data buffers.
                                           036F   758 ;
                                           036F   759
                     0A A6   46 8F   90    036F   760          MOVB    #DYN$C_PMB,PMB$B_TYPE(R6) ; Set structure type to PMB
                                           0374   761
                                           0374   762          ASSUME  PMB$Q_SBPHDR EQ PMB$Q_HDR+8
                                           0374   763          ASSUME  PMB$L_ASTQFL EQ PMB$Q_SBPHDR+8
                                           0374   764
                        50   14 A6   9E    0374   765          MOVAB   PMB$Q_HDR(R6),R0           ; get queue header address
                                60   60   DE  0378   766          MOVAL   (R0),(R0)                  ; init empty queue flink
                                80   80   DE  037B   767          MOVAL   (R0)+,(R0)+                ; init empty queue blink
                                60   60   DE  037E   768          MOVAL   (R0),(R0)                  ; init subprocess queue flink
                                80   80   DE  0381   769          MOVAL   (R0)+,(R0)+                ; init subprocess queue blink
                                60   60   DE  0384   770          MOVAL   (R0),(R0)                  ; init AST queue flink
                                60   80   DE  0387   771          MOVAL   (R0)+,(R0)                 ; init AST queue blink
                                           038A   772
                                           038A   773          ASSUME  PMB$L_BUFBASE EQ PMB$L_CURBUF+4
                                     66   7C  038A   774          CLRQ    PMB$L_CURBUF(R6)           ; note no current buffer
                                           038C   775
                                  0B A6   94   038C   776          CLRB    PMB$B_FLAGS(R6)            ; insure that all flags are clear
                                  10 A6   D4   038F   777          CLRL    PMB$L_OVERFLOW(R6)         ; indicate no overflows
                           30 A6  60 A4   D0   0392   778          MOVL    PCB$L_PID(R4),PMB$L_PID(R6) ; Insert PID into PMB/ACB
                              0C A6   01   CE   0397   779          MNEGL   #1,PMB$L_LASTCPU(R6)       ; force timestamping of 1st record
                                           039B   780
                                  59   05   3C   039B   781          MOVZWL  #BUFCNT,R9                 ; number of data buffers to allocate
                           51   020C 8F   3C   039E   782 10$:      MOVZWL  #BUFSIZ,R1                 ; get size of buffer to allocate
                            00000000'EF   16   03A3   783          JSB     EXE$ALLOCBUF               ; allocate buffer
                                  0F   50   E9   03A9   784          BLBC    R0,20$                     ; take error path if no memory available
                        0A A2     47   8F   90   03AC   785          MOVB    #DYN$C_PFB,PFB$B_TYPE(R2) ; Set structure type to PFB
                              14 A6     62   0E   03B1   786          INSQUE  (R2),PMB$Q_HDR(R6)         ; queue on empty buffer list
                                  E6   59   F5   03B5   787          SOBGTR  R9,10$                     ; back for more buffers
                                           03B8   788
                                  6E   01   3C   03B8   789          MOVZWL  #SS$_NORMAL,(SP)           ; Indicate success
                                       50 8ED0   03BB   790 20$:      POPL    R0                         ; Return w/status in R0
                                          05   03BE   791          RSB
                                           03BF   792
```

```
                                        .SBTTL  PFM$MON - Resident Monitoring Code

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Resident code called by memory management to record
                                ;       page fault PC and VA.  Data is inserted into a buffer.
                                ;       When the buffer is full, it is queued for a cooperating
                                ;       process which outputs the data to disk.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSB/JSB PFM$MON
                                ;
                                ; INPUTS:
                                ;
                                ;       R4 = PCB address
                                ;       R5 = PHD address
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       none
                                ;
                                ; OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       none
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;
                                ; ENVIRONMENT:
                                ;
                                ;       Kernel mode, IPL = IPL$_SYNCH
                                ;--
                                
                                        .PSECT  AEXENONPAGED,LONG
                                        .ENABLE         LSB
02  0C A2   D1  0000  5$:       CMPL    PFB$L_RECCNT(R2),#2     ; Space for time stamp and PC/VA pair ?
      0C   1E  0004              BGEQU   6$                      ; BR if space in buffer
      55   DD  0006              PUSHL   R5                      ; Save PHD address
      61   10  0008              BSBB    FLUSH_BUFFER_INT        ; Flush buffer (R2, R4, R5 DESTROYED)
      55 8ED0  000A              POPL    R5                      ; Restore PHD address
    0093   30  000D              BSBW    GETBUF                  ; try to get next buffer
      49   1D  0010              BVS     30$                     ; exit if none there, lose data
  81  01   CE  0012      6$:     MNEGL   #1,(R1)+                ; Add time stamp to buffer
```

```
                     0015   851          SETIPL  #IPL$_HUCLK                    ; Synchronize with CPU time updating
          81  38 A5  D0 0018   852          MOVL    PHD$L_CPUTIM(R5),(R1)+
              38 A5  D0 001C   853          MOVL    PHD$L_CPUTIM(R5),-          ; Update reference CPU time
                 0C A6       001F   854                  PMB$L_LASTCPU(R6)
                             0021   855          SETIPL  #IPL$_SYNCH             ; Lower IPL to entry IPL
                 0C A2  D7   0024   856          DECL    PFB$L_RECCNT(R2)        ; Adjust record count for time stamp
                    1C  11   0027   857          BRB     15$                     ; Rejoin mainline code
                             0029   858
                             0029   859          .ALIGN  LONG
                             002C   860  PFM$MON::
                 007F  8F    BB 002C   861          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6> ; save registers
          56  011C  C4    D0 0030   862          MOVL    PCB$L_PMB(R4),R6        ; get PMB address
                             0035   863
                             0035   864          ASSUME  PMB$L_BUFBASE   EQ      PMB$L_CURBUF+4
                             0035   865
              51     66    7D 0035   866          MOVQ    PMB$L_CURBUF(R6),R1     ; get buffer address & base
                        04  12 0038   867          BNEQ    10$                     ; branch if buffer exists
                        67  10 003A   868          BSBB    GETBUF                  ; try to get next buffer
                        1D  1D 003C   869          BVS     30$                     ; exit if none there, lose data
      0C A6        38 A5  D1 003E   870  10$:      CMPL    PHD$L_CPUTIM(R5),PMB$L_LASTCPU(R6) ; Need a CPU time stamp ?
                        BB  12 0043   871          BNEQ    5$                      ; BR if yes
          81      40 AE    D0 0045   872  15$:     MOVL    FAULTPC(SP),(R1)+       ; insert pc of instruction
          81      3C AE    D0 0049   873          MOVL    FAULTVA(SP),(R1)+       ; insert va which faulted
                66     51    D0 004D   874          MOVL    R1,PMB$L_CURBUF(R6)     ; save current buffer address
                   02 0C A2  F5 0050   875          SOBGTR  PFB$L_RECCNT(R2),20$    ; Buffer full ?
                        15  10 0054   876          BSBB    FLUSH_BUFFER_INT        ; Yes, flush it
                             0056   877                                          ; (R2, R4, R5 DESTROYED)
                 007F  8F    BA 0056   878  20$:     POPR    #^M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
                        05  005A   879          RSB
                    10 A6    D6 005B   880  30$:     INCL    PMB$L_OVERFLOW(R6)      ; Count an overflow
                        F6  11 005E   881          BRB     20$
                             0060   882          .DISABLE        LSB
                             0060   883
                             0060   884          .ENABLE         LSB
                             0060   885
          0C A2      3F    D0 0060   886  5$:      MOVL    #MAXREC,PFB$L_RECCNT(R2); Re-initialize records remaining
                        05  0064   887          RSB                             ; and return
                             0065   888
                             0065   889                                          ; (R2, R4, R5 DESTROYED)
                             0065   890  FLUSH_BUFFER:                          ; Entry with IPL=IPL$_ASTDEL
          52      04 A6    D0 0065   891          MOVL    PMB$L_BUFBASE(R6),R2    ; Get current buffer base address
                        25  13 0069   892          BEQL    20$                     ; Just exit if no current buffer
                             006B   893  FLUSH_BUFFER_INT:                      ; Entry with IPL=IPL$_SYNCH, R2 set
          3F    0C A2      C3 006B   894          SUBL3   PFB$L_RECCNT(R2),#MAXREC,- ; Convert records remaining to
                0C A2         006F   895                  PFB$L_RECCNT(R2)        ; number of records in buffer
                        ED  13 0071   896          BEQL    5$                      ; BR if no records in buffer
                             0073   897          ASSUME  PFB$L_FLINK EQ 0
          20 B6      62    0E 0073   898          INSQUE  (R2),@PMB$Q_SBPHDR+4(R6) ; insert buffer at end of write queue
                             0077   899          ASSUME  PMB$V_MODE EQ 0
                             0077   900          ASSUME  PMB$K_IMAGE EQ 1
             16 0B A6    E9 0077   901          BLBC    PMB$B_FLAGS(R6),30$     ; BR if subprocess mode
      0E 0B A6    01    E2 007B   902          BBSS    #PMB$V_ASTIP,PMB$B_FLAGS(R6),10$ ; If AST in progress, return
          0B A6    04    88 0080   903          BISB2   #PMB$M_QAST,PMB$B_FLAGS(R6) ; Show embedded ACB queued on PCB
             52    01    9A 0084   904          MOVZBL  #PRI$_IOCOM,R2          ; R2 <= priority increment class
          55    24 A6    DE 0087   905          MOVAL   PMB$L_ASTQFL(R6),R5     ; R5 <= address of ACB
                FF72'      30 008B   906          BSBW    SCH$QAST                ; Signal full buffer available
                   66      7C 008E   907  10$:     CLRQ    PMB$L_CURBUF(R6)        ; Note that no current buffer exists
```

```
                05  0090    908 20$:    RSB
                    0091    909
      51  30 A6    D0  0091    910 30$:    MOVL    PMB$L_PID(R6),R1        ; get params to wake up other process
                    0095    911         DSBINT  #IPL$_SYNCH
           FF62'    30  0098    912         BSBW    SCH$WAKE                ; and wake the process
                    009E    913         ENBINT
             EB     11  00A1    914         BRB     10$
                    00A3    915
                    00A3    916         .DISABLE        LSB
                    00A3    917
                    00A3    918 GETBUF:                                     ; get next buffer from queue
      52  14 B6    0F  00A3    919         REMQUE  @PMB$Q_HDR(R6),R2       ; dequeue next buffer
            14     1D  00A7    920         BVS     10$                     ; leave with V set if none there
   04 A6    52     D0  00A9    921         MOVL    R2,PMB$L_BUFBASE(R6)    ; save buffer base address in PMB
      51  14 A2    9E  00AD    922         MOVAB   PFB$B_BUFFER(R2),R1     ; skip buffer overhead
   0C A2     3F     D0  00B1    923         MOVL    #MAXREC,PFB$L_RECCNT(R2); Initialize records remaining
10 A2     10 A6    D0  00B5    924         MOVL    PMB$L_OVERFLOW(R6),PFB$L_OVERFLOW(R2) ; Copy overflow count
          10 A6    D4  00BA    925         CLRL    PMB$L_OVERFLOW(R6)      ; and then reset it (also clears V)
                    00BD    926                                             ; (R1 & R2 point to buffer)
                05  00BD    927 10$:    RSB                                 ; return with V set or clear
                    00BE    928
                    00BE    929         .END
```

Symbol table

```
$$T1                    = 00000001
ACB$M_NODELETE          = 00000020
ACB$M_PKAST             = 00000010
ACMODE                  = 00000010
ALLPMB                    0000033B R      02
ASTADR                  = 00000008
ASTPRM                  = 0000000C
BUFCNT                  = 00000005
BUFSIZ                  = 0000020C
CHECK_ACMODE              000000AC R      02
CHECK_STOP                000000CE R      02
CTL$GL_PHD                ******** X      02
DASSGN_EXIT               0000025D R      02
DEQUEUE_ACB               000002D6 R      02
DYN$C_PFB               = 00000047
DYN$C_PMB               = 00000046
EXE$AB_HEXTAB             ******** X      02
EXE$ALLOCBUF              ******** X      02
EXE$DEANONPAGED           ******** X      02
EXE$EPID_TO_IPID          ******** X      02
EXE$IPID_TO_PCB           ******** X      02
EXE$MAXACMODE             ******** X      02
EXE$SETPFM                0000006A RG     02
FAULTPC                 = 00000040
FAULTVA                 = 0000003C
FILWRT                    00000043 R      02
FILWRTPRV                 00000052 R      02
FLUSH_BUFFER              00000065 R      03
FLUSH_BUFFER_INT          0000006B R      03
GETBUF                    000000A3 R      03
ILLSEQOP                  00000061 R      02
IO$_READVBLK            = 00000031
IPL$_ASTDEL             = 00000002
IPL$_HWCLK              = 00000018
IPL$_SYNCH              = 00000008
JIB$L_BYTCNT            = 00000020
MAXREC                  = 0000003F
NOPRIV                    000000BE R      02
PCB$L_EPID              = 00000064
PCB$L_JIB               = 00000080
PCB$L_PID               = 00000060
PCB$L_PMB               = 0000011C
PFB$B_BUFFER            = 00000014
PFB$B_TYPE              = 0000000A
PFB$B_USER_BUFFER       = 0000000C
PFB$C_LENGTH            = 0000020C
PFB$L_FLINK             = 00000000
PFB$L_OVERFLOW          = 00000010
PFB$L_RECCNT            = 0000000C
PFB$S_USER_BUFFER       = 00000200
PFM$C_BUFCNT            = 00000005  G
PFM$GETBUF                000002EC RG     02
PFM$MON                   0000002C RG     03
PFM$PURGE                 0000026A RG     02
PFMBUSY                   0000005A R      02
PFMFILWRT                 00000000 R      02
PFMFLG                  = 00000004
PFMQUOTA                  00000020 R      02
PFM_PKAST                 000002D1 R      02
PHD$L_CPUTIM            = 00000038
PHD$V_PFMFLG            = 00000000
PHD$W_FLAGS             = 00000036
PMB$B_ACMODE            = 0000002E
PMB$B_FLAGS             = 0000000B
PMB$B_RMOD              = 0000002F
PMB$B_TYPE              = 0000000A
PMB$C_LENGTH            = 00000040
PMB$K_IMAGE             = 00000001
PMB$L_AST               = 00000034
PMB$L_ASTPRM            = 00000038
PMB$L_ASTQFL            = 00000024
PMB$L_BUFBASE           = 00000004
PMB$L_CURBUF            = 00000000
PMB$L_EPID              = 00000034
PMB$L_KAST              = 0000003C
PMB$L_LASTCPU           = 0000000C
PMB$L_OVERFLOW          = 00000010
PMB$L_PID               = 00000030
PMB$M_ASTIP             = 00000002
PMB$M_QAST              = 00000004
PMB$Q_HDR               = 00000014
PMB$Q_SBPHDR            = 0000001C
PMB$V_ASTIP             = 00000001
PMB$V_MODE              = 00000000
PMB$V_QAST              = 00000002
PMB$W_MBXCHN            = 0000002C
PMB$W_SIZE              = 00000008
PQL$_BYTLM              = 00000003
PQL$_CPULM              = 00000004
PQL$_FILLM              = 00000006
PQL$_LISTEND            = 00000000
PQL$_PGFLQUOTA          = 00000007
PQL$_PRCLM              = 00000008
PQL$_TQELM              = 00000009
PR$_IPL                 = 00000012
PRI$_IOCOM              = 00000001
PSL$S_PRVMOD            = 00000002
PSL$V_PRVMOD            = 00000016
PURGE_EXIT                00000229 R      02
PURGE_EXIT_RO             0000022C R      02
PURGE_EXIT_RO_VEC         000000C8 R      02
PURGE_EXIT_VEC            000000CB R      02
QUOTA_CHARGE            = 00000A7C
SCH$GL_CURPCB             ******** X      02
SCH$QAST                  ******** X      03
SCH$REMOVACB              ******** X      02
SCH$WAKE                  ******** X      03
SET_ASTMODE               00000116 R      02
SET_SUBPMODE              00000130 R      02
SS$_BUFFEROVF           = 00000601
SS$_EXQUOTA             = 0000001C
SS$_ILLSEQOP            = 000002DC
SS$_INSFMEM             = 00000124
SS$_NODATA              = 000001AC
```

```
SS$_NOPRIV                      = 00000024
SS$_NORMAL                      = 00000001
SS$_PFMBSY                      = 00000204
START                             000000E3 R      02
START_OK                          0000010D R      02
STOP                              0000020A R      02
STOP_PROCESS                      0000022F R      02
STOP_VEC                          000000C5 R      02
SUBFUNC                           00000090 R      02
SYS$CREMBX                        ********  GX    02
SYS$CREPRC                        ********  GX    02
SYS$DASSGN                        ********  GX    02
SYS$FORCEX                        ********  GX    02
SYS$GETCHN                        ********  GX    02
SYS$QIOW                          ********  GX    02
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YEXEPAGED	000003BF (959.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
AEXENONPAGED	000000BE (190.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-------------------------+
! Performance indicators !
+-------------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.26
Command processing	106	00:00:00.55	00:00:01.73
Pass 1	407	00:00:15.76	00:00:35.84
Symbol table sort	0	00:00:02.35	00:00:05.76
Pass 2	165	00:00:03.46	00:00:08.12
Symbol table output	16	00:00:00.14	00:00:00.22
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	727	00:00:22.36	00:00:51.95

```
The working set limit was 1500 pages.
88818 bytes (174 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1517 non-local and 40 local symbols.
929 source lines were read in Pass 1, producing 20 object records in Pass 2.
38 pages of virtual memory were used to define 36 macros.
```

```
                                +----------------------------+
                                ! Macro library statistics !
                                +----------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      13
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   19
TOTALS (all libraries)                              32

1718 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSSETPFM/OBJ=OBJ$:SYSSETPFM MSRC$:SYSSETPFM/UPDATE=(ENH$:SYSSETPFM)+EXECML$/LIB
```

SYSSETPRA LIS
SYSSETSSF LIS
SYSSNDJBC LIS
SYSSETIME LIS
SYSSETPFM LIS
SYSUPDSEC LIS
SYSSNDMSG LIS
SYSSETPRT LIS
SYSSETPRV LIS
SYSUNWIND LIS
SYSSETSTK LIS
SYSSETMOD LIS
SYSSETEXV LIS
SYSSETPRI LIS
SYSSCHEVT LIS